The SURE Approach to Reliability Analysis

Ricky W. Butler
NASA Langley Research Center, Hampton

Key Words — **Markov analysis, Fault-tolerant system, Modeling, Semi-Markov**

Reader Aids —
Purpose: Advance state of the art
Special math needed for explanation: Markov concepts
Special math needed to use results: Elementary probability
Results useful to: Reliability analysts and theoreticians

Summary & Conclusions — **The SURE computer program is a reliability-analysis tool for ultrareliable computer-system architectures. SURE is based on computational methods developed at the NASA Langley Research Center. These methods provide an efficient means for computing reasonably accurate upper and lower bounds for the death state probabilities of a large class of semi-Markov models. Once a semi-Markov model is described using a simple input language, SURE automatically computes the upper and lower bounds on the probability of system failure. A parameter of the model can be specified as a variable over a range of values, thus directing SURE to perform a sensitivity analysis automatically. This feature, along with the speed of the program, makes it an especially useful design tool.**

SURE is a flexible, user-friendly reliability-analysis tool. The program provides a rapid computational capability for semi-Markov models useful for describing the fault-handling behavior of fault-tolerant computer systems. The only modeling restriction imposed by the program is that the non-exponential recovery transitions must be fast in comparison to the mission time – a desirable attribute of all fault-tolerant systems. The SURE reliability-analysis method uses a fast bounding theorem based on means and variances; the method yields upper and lower bounds on the probability of system failure. The upper and lower bounds are typically within 5 percent of each other. Techniques have been developed to enable SURE to solve models with loops and calculate the operational-state probabilities. The computation is extremely fast, and large state-spaces can be directly solved; a pruning technique enables SURE to process extremely large models.

1. INTRODUCTION

The SURE computer program is a general purpose reliability analysis tool especially useful for the analysis of fault-tolerant digital computer systems. The first version of the program was developed in 1983. Over the past 7 years the capabilities of the program have been increased to handle larger and more complex systems. This paper overviews SURE with an emphasis on its solution techniques.

SURE was developed in response to the growing size and complexity of fault-tolerant digital systems and the resulting intractable reliability analysis. Because of the importance of the reliability analysis of these systems, many mathematical approaches have been developed during the 1980s to deal with these systems. Many of these approaches have been incorporated into automated reliability-analysis tools. Some of the most widely known are CARE III [1], HARP [2], SHARPE [3], SURF [4], and ARIES [5]. Johnson & Malek [6] surveyed many of these tools.

SURE consists of about 4000 lines of Pascal code. It runs on VMS and Unix operating systems. It is now distributed by COSMIC, the US Government software distribution center. The source files can be purchased from COSMIC by calling 404-542-3265. The COSMIC code-number for SURE is LAR-13789.

SURE is based on a mathematical theorem developed by White [7, 8] which provides a method for computing the reliability of a fault-tolerant system. Two characteristics of a fault-tolerant system have traditionally made this task difficult.

- The use of sophisticated reconfiguration strategies has resulted in complex models.
- System recovery is many orders of magnitude faster than the fault-arrival process. This causes rapid growth in the error terms in numerical integration algorithms. □

The mathematical theorem for SURE solves both of these problems for systems with slow fault-arrival processes and fast system-recovery (viz, a well-designed fault-tolerant system). The theorem establishes that just the means and variances of the recovery times are sufficient information about the reconfiguration process in order to obtain tight bounds on the probability of system failure. The bounds consist of an algebraic factor using the means and variances of the system recovery times and a factor that is the solution of a numerically stable differential equation whose coefficients are the slow fault-occurrence rates. The differential equation is tractable enough that for a many cases its solution has easy algebraic upper and lower bounds. (SURE automatically selects the appropriate method and informs the user when the differential-equation option is used.) Thus, the bounding theorem reduces the traditionally difficult problem to easily computed mathematics.

Unlike the other tools developed under NASA Langley sponsorship (CARE III and HARP), SURE does not use behavioral decomposition to gain computational efficiency. Instead, the mathematical bounding theorem provides upper and lower bounds on system probability of failure in terms of an easily computed algebraic formula. The use of strict mathematical bounds avoids some of the problems associated with approximations used in earlier tools [9, 10]. The program does not use a separate *fault-handling* model to deal with system reconfiguration. Since the mathematical theorem encompasses the class of semi-Markov models, a complex reconfiguration process can be captured in one general recovery transition. Both CARE III and HARP are based upon a *critical-pair* approach wherein 2 coincident faults cause system failure. However, SURE can solve models where three or more faults must be

0018-9529/92$03.00

present before system failure occurs. In fact, SURE is not limited to any particular modeling philosophy or graphical structure. The upper and lower bounds produced by SURE will be close as long as the non-exponential (recovery) transitions are fast compared with the exponential (failure) transitions. The primary limitation of SURE is that slow transitions must be exponentially distributed. Thus, systems having non-exponential failure characteristics can not be directly analyzed. However, globally time-dependent failure behavior (nonhomogeneous) can be analyzed using piece-wise linear upper and lower bounds on the globally time-dependent failure distribution [21]. This technique is appreciably slower than the usual solution techniques because it requires a manual iterative analysis of the model over a sequence of steps.

Section 2 overviews the techniques to develop a semi-Markov model of a fault-tolerant computer system. Section 3 presents the basic mathematics of SURE. Section 4 describes the SURE user-interface, with a sample interactive session. Section 5 gives the basis for the model-pruning capability. Section 6 justifies the SURE loop-truncation method. Section 7 describes some miscellaneous additional features.

Notation

1. Greek letters represent the rates of exponential transitions and Roman letters represent the Cdf's of the fast recovery transitions.

2. Standard notation is given in "Information for Readers & Authors" at the rear of each issue.

2. RELIABILITY MODELING OF FAULT-TOLERANT COMPUTER ARCHITECTURES

2.1 *Modeling the State Transitions*

Highly reliable systems use *parallel* redundancy to achieve their fault tolerance since current manufacturing techniques cannot produce circuitry with adequate reliability. Redundant processing and voting are used to mask the errors produced by a failed component. Reconfiguration can increase the reliability of the system without the overhead of even more redundancy. Reconfigurable systems exhibit behavior that involves both slow and fast processes, and, when modeled stochastically, some state transitions are many orders of magnitude faster than others. The slower transitions correspond to fault arrivals in the system. The faster transition rates correspond to system recovery from faults.

If the system states are delineated appropriately, the slow transitions can be obtained from field data and/or by using Mil-Hdbk-217 [11]. The transition rates are usually assumed to be reasonably constant (exponential distribution of life) during the useful lifetime for many electronic devices [12]. The system recovery processes can be measured experimentally using fault injection. In a pure Markov model, the recovery process is typically represented as one constant-rate transition. However, experiments on the Fault-Tolerant Multiprocessor computer architecture have demonstrated [13] that these transition rates are not necessarily constant. Some tools have provided detailed multi-step fault-handling models to capture this non-exponential behavior [1, 14]. Since SURE solves semi-Markov models, the reconfiguration process can be modeled directly with one general recovery transition. Furthermore, since the mathematical bounds depend only upon the means and standard deviations of the recovery transitions, distribution fitting is unnecessary. Given an empirical distribution of system recovery times, the easily calculated sample means and standard deviations can be used directly.

2.2 *Example 1*

Figure 1 shows a semi-Markov model of 3 processors with 1 spare.

Assumptions & Notation

1. The outputs of the processors use 3-way voting to mask faults.

2. The spare does not fail while inactive (cold standby).

3. Processor fault arrivals occur with constant transition-rate, λ.

4. Recovery #1 is by replacing the faulty processor with a spare.

5. Recovery #2 is by degrading to a simplex processor.

6. The recovery #1 & #2 processes are different. The Cdf's of recovery-time are $F_1(t)$ and $F_2(t)$.

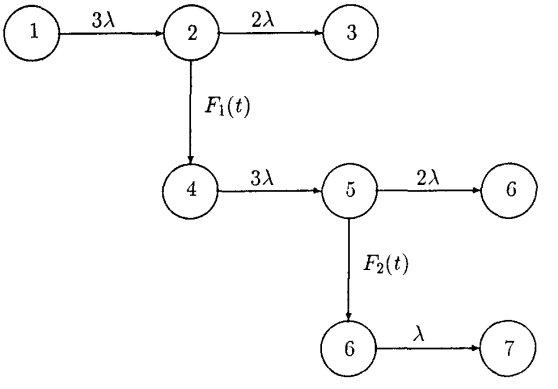


Figure 1. Semi-Markov Model of a Triad with One Spare

The horizontal transitions represent fault arrivals. The coefficients of λ represent the number of processors in the configuration that can fail. The vertical transitions represent recovery from a fault. Since the system uses 3-way voting for fault masking, there is a race between the occurrence of fault #2 and the removal of fault #1. If fault #2 wins the race, then the system fails (state 3).

The input language to SURE is very simple. The input model is defined by listing all of the transitions of the model. This model is defined as:

Identifiers

```
1. LAMBDA = 1E-4;    (* Failure rate of a processor *)
2. MU1 = 2.7E-4;     (* Mean time to replace faulty processor w/ a spare *)
3. SIGMA1 = 1.4E-4;  (* Standard deviation of time to replace w/ a spare *)
4. MU2 = 9.2E-4;     (* Mean time to degrade to a simplex *)
5. SIGMA2 = 3.8E-4   (* Standard deviation of time to degrade to simplex *)
```

Transition Statements

```
1,2 = 3*LAMBDA;
2,3 = 2*LAMBDA;
2,4 = <MU1,SIGMA1>;
4,5 = 3*LAMBDA;
5,6 = 2*LAMBDA;
5,7 = <MU2,SIGMA2>;
7,8 = LAMBDA;
```

The first 5 statements equate values to identifiers (symbolic names). Identifier #1, LAMBDA represents the processor failure rate. Identifiers #2 & #3, MU1 & SIGMA1 are the mean and standard deviation of the time to replace a faulty processor with a spare. Identifiers #4 & #5, MU2 & SIGMA2 are the mean and standard deviation of the time to degrade to a simplex. Conveniently, the only information SURE needs about the non-exponential recovery processes are the means and standard deviations. The final 7 statements define the transitions of the model. If the transition is a slow fault-arrival then only the exponential rate need be provided. For example, the last statement defines a transition from state 7 to state 8 with rate LAMBDA. If the transition is a fast recovery then the mean and standard deviation of the recovery time must be given. For example, the statement: 2,4 = <MU1,SIGMA1> defines a transition from state 2 to state 4 with mean recovery time, MU1 and standard deviation SIGMA1.

2.3 *General Modeling*

The development of a reliability model of a complex system uses the same concepts used in the development of the model in section 2.2. The two types of transitions (failure and recovery) are still used, but there often are many types of failure and different recoveries for each type. Therefore, there can be several failure transitions from a state, each representing a failure of a different part of the system. Likewise, some states are reached after a sequence of different failures, and there are multiple recoveries from the state.

3. THE FUNDAMENTAL SURE MATHEMATICS

This section presents the White semi-Markov bounding theorem upon which SURE is based. Some notation is developed; then the details of the theorem are presented.

3.1 *Path-Step Classification & Notation*

The theorem provides bounds on the death-state probabilities at a specified time. It assumes that the system is initially in a single state – the start-state. (The generalization to multiples states is in section 6.1.) SURE finds every path from the start-state to a death-state. Each path's contribution to system failure is calculated separately using the White semi-Markov bounding theorem.

Let each state along the path be put into 1 of 3 classes which are distinguished by the type of transitions leaving the state. A state and all the transitions leaving it is a *path-step*. The transition on the path being analyzed is the *on-path transition*. In the figures in this section, the on-path transition is always the horizontal transition. (This is different from figure 1 where the horizontal transitions were fault arrivals and vertical transitions were recoveries.) The remaining transitions are *off-path transitions*. The classification is made on the basis of whether the on-path and off-path transitions are slow (and hence with constant transition rate) or fast. If there are no off-path transitions, the path-step is classified as if it contained a slow off-path transition. Thus, the following classes of path-steps are of interest.

Class-1: Slow On-Path, Slow Off-Path

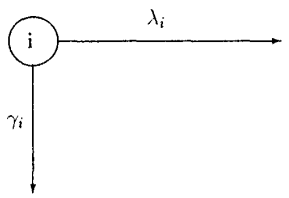


Figure 2. Class 1: Slow On-Path, Slow Off-Path

Notation

λ_i on-path constant transition-rate
γ_i sum of slow off-path transition rates □

If all transitions leaving the state are slow, then the path-step is class-1. There can be an arbitrary number of slow off-path transitions. If any of the off-path transitions are not slow, then the path-step is class-3. The path-steps $1 \rightarrow 2$, $4 \rightarrow 5$ and $5 \rightarrow 6$ in the triad-plus-1-spare model of figure 1 are examples. □

Class-2: Fast On-Path, Arbitrary Off-Path

Notation

ϵ_i sum of all slow off-path transition rates
$F_{i,k}$ Cdf of fast transition k from state i

If the on-path transition is fast then the path-step is class 2; see figure 3. There can be an arbitrary number of slow or fast off-path transitions. As before, the off-path slow, constant-rate transitions can be represented as a single transition with

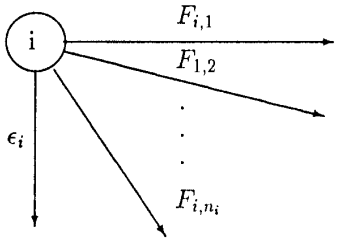


Figure 3. Class 2: Fast On-path, Arbitrary Off-Path

a rate ϵ_i equal to the sum of all the slow off-path transition rates. The path-step 2 → 4 in the triad-plus-1-spare model of figure 1 are examples. The distribution of the fast on-path transition is $F_{i,1}$. The Cdf of time for fast transition k from state i is $F_{i,k}$. Three measurable parameters must be specified for each fast transition (given that this transition occurs):

- transition probability, $\rho(F^*_{i,k}) \equiv \int_0^\infty \Psi_{i,k}(t)dF_{i,k}(t)$
- conditional mean, $\mu(F^*_{i,k})$

$$\equiv [\rho(F^*_{i,k})]^{-1} \cdot \int_0^\infty t \cdot \Psi_{i,k}(t)dF_{i,k}(t)$$

- conditional variance, $\sigma^2(F^*_{i,k})$

$$\equiv [\rho(F^*_{i,k})]^{-1} \cdot \int_0^\infty t^2 \cdot \Psi_{i,k}(t)dF_{i,k}(t) - \mu^2(F^*_{i,k})$$

$$\Psi_{i,k}(t) \equiv \prod_{j \neq k} \overline{F_{i,j}}(t),$$

conditional implies the condition, "Given that the transition occurs."

These population parameters are measured by the —

- sample fraction of times that a fast transition is successful,
- mean of the sample that is conditional on the transition's occurring,
- variance of the sample that is conditional on the transition's occurring.

In any experiment where competing processes are studied, the observed empirical distributions are conditional on the transition's occurring. The time it takes a system to transit to the next state is observed only when that transition occurs. The asterisk denotes that the parameters are defined in terms of the conditional distributions. These expressions are defined independently of the "exponential" transitions ϵ_j. Consequently, the sum of the fast-transition probabilities $\Sigma_{i,k}\, \rho(F^*_{i,k}) = 1$. In particular, if there is only one fast transition, its probability is 1 and the conditional population-mean is equivalent to the unconditional population-mean. (The SURE user does not have to deal explicitly with the unconditional distributions $F_{i,k}$. However, in order to develop the mathematical theory, they must be used.)□

Class-3: Slow On-Path, Fast Off-Path

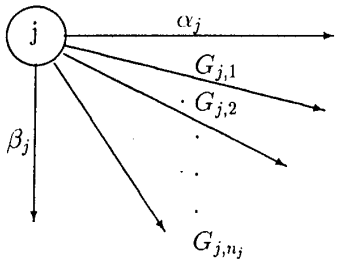


Figure 4. Class 3: Slow On-Path, Fast Off-Path

Notation

α_j	slow on-path transition rate
β_j	sum of the slow off-path transition rates
$G_{j,k}$	Cdf of fast off-path transition-time from state j to state k
H_j	Cdf of *recovery holding time* in state j □

The on-path transition must be slow in order for a path-step to be categorized as class-3. There can be both slow and fast off-path transitions, but at least one off-path transition must be fast; see figure 4. The path-step 2 → 3 in the triad-plus-1-spare model of figure 1 are in this class. As in class-2, the transition probability $\rho(G^*_{j,k})$, the conditional mean $\mu(G^*_{j,k})$, and the conditional variance $\sigma^2(G^*_{j,k})$ must be given for each fast off-path transition. There really is no difference between transitions labeled with the letters F and G. The different letters are used to help keep track of the context, viz, whether the transition is class-2 (labeled F) or class-3 (labeled G) in the current path. In either case, the SURE user supplies the conditional mean, the conditional standard deviation, and the transition probability.

Although, these 3 parameters suffice to specify a class-3 path-step to SURE, the mathematical theory is more easily expressed in terms of the *holding time in the state*, viz, time the system remains in the state before it transits to some other state. The bounding theorem is expressed using a slightly different form of holding time which, to prevent confusion, is referred to as *recovery holding time*, viz, holding time in the state with the slow exponential distributions removed from the state. Since the slow exponential transition rates are many orders of magnitude smaller than the fast transition rates, the recovery holding time is approximately equal to the holding time in the state. The following Cdf and parameters are used in the theorem:

$$H_j(t) = 1 - \prod_{k=1}^{n_j} \overline{G_{j,k}}(t),$$

$$\mu(H_j) = \int_0^\infty \overline{H_j}(t)dt,$$

$$\sigma^2(H_j) = 2\cdot\int_0^\infty t\cdot\bar{H}_j(t)dt - \mu^2(H_j).$$

These 2 parameters are the mean and variance of the holding time in state j without consideration to the slow exponential transitions (viz, with the slow exponential transitions removed). These 2 parameters do not have to be supplied to SURE; SURE derives them from the other inputs - $\rho(G^*_{j,k})$, $\mu(G^*_{j,k})$, $\hat{\sigma^2}(G^*_{j,k})$ [these 3 parameters are defined exactly the same way as the class-2 path-step parameters] - as follows:

$$\mu(H_j) = \sum_{k=1}^{n_j} \rho(G^*_{j,k})\cdot\mu(G^*_{j,k}),$$

$$\sigma^2(H_j) = \left[\sum_{k=1}^{n_j} \rho(G^*_{j,k})\cdot[\sigma^2(G^*_{j,k}) + \mu^2(G^*_{j,k})]\right] - \mu^2(H_j).$$

Although, the fast distributions are specified without considering the competing slow exponential transitions, the theorem gives bounds that are correct in the presence of such exponential transitions. The parameters are defined in this manner to simplify the process of specifying a model. Throughout the paper, the *holding time in a state* in which the slow transitions have been removed from the state is referred to as *recovery holding time*.

For convenience, when referring to a specific path in the model, the Cdf of an on-path fast transition is indicated by a single subscript which specifies the source state. For example, if the transition with Cdf $F_{j,k}$ is the on-path transition, then it can be referred to as F_j.

Notation

$F_{j,k}$	fast transition k from state j
F_j	on-path fast transition from state j □

3.2 *SURE Bounding Theorem*

Notation

T	mission time
k	number of class-1 path-steps
m	number of class-2 path-steps
n	number of class-3 path-steps □

Theorem: The probability $D(T)$ of entering a particular death-state within the mission time T, following a path with k class-1 path-steps, m class-2 path-steps, and n class-3 path-steps, is bounded as follows:

$$\text{LB} < D(T) < \text{UB},$$

$$\text{UB} \equiv Q(T)\cdot\prod_{i=1}^{m}\rho(F^*_i)\cdot\left[\prod_{j=1}^{n}\alpha_j\cdot\mu(H_j)\right],$$

$$\text{LB} \equiv Q(T-\Delta)\cdot\prod_{i=1}^{m}[\rho(F^*_i)\cdot Z_i]\cdot\left[\prod_{j=1}^{n}\alpha_j\cdot Y_j\right],$$

for $T-\Delta > 0$, for all $r_i > 0$, for all $s_j > 0$,

$$\Delta \equiv \sum_{i=1}^{m} r_i + \sum_{j=1}^{n} s_j,$$

$$Y_j \equiv \mu(H_j) - [\tfrac{1}{2}\cdot(\alpha_j+\beta_j) + (1/s_j)]\cdot\omega^2(H_j)$$

$$Z_i \equiv 1 - \epsilon_i\cdot\mu(F^*_i) - \omega^2(F^*_i)/r_i^2$$

$$\omega^2(z) \equiv \mu^2(z) + \sigma^2(z), \text{ mean-square deviation for } z$$

$$Q(x) \equiv \Pr\{\text{traversing a path consisting of only the class-1 path-steps within time } x\}.$$ □

Proof: See [7, 15].

Different choices of these parameters lead to different bounds; SURE uses:

$$r_i^3 = 2\cdot T\cdot\omega^2(F^*_i),$$

$$s_j^2 = T\cdot\omega^2(H_j)/\mu(H_j),$$

which give very close (very near optimal) bounds in practice [15].

Two simple algebraic approximations for $Q(T)$ are [16]:

$$Q(T) < Q_u(T) = \left(\prod_{i=1}^{k}\lambda_i\cdot T\right)/k!$$

$$Q(T) > Q_l(T) = Q_u(T)\cdot[1 - w_k/(k+1)]$$

$$w_k \equiv \sum_{i=1}^{k}(\lambda_i+\gamma_i)\cdot T, \text{ mean total time in class-1 transitions}$$

Both $Q_u(T)$ & $Q_l(T)$ are close to $Q(T)$ as long as $w_k \ll 1$, viz, mission time is short compared to the average component-lifetime.

SURE uses the following slightly improved upper bound on $Q(T)$:

$$Q(T) < Q^*_u(T) = \left(\prod_{i\in S}\lambda_i\cdot T\right)/|S|!, \text{ for } S \equiv \{i|\lambda_i\cdot T < 1\}.$$

$Q^*_u(T)$ is obtained by removing all the fast exponential transitions from $Q_u(T)$. Since the path is shorter, the probability of reaching the death-state is larger than for $Q_u(T)$ model. $Q^*_u(T)$ & $Q_l(T)$ are used for the QTCALC=0 option. For the QTCALC=1 option, a differential-equation-solver calculates $Q(T)$ & $Q(T-\Delta)$. For the QTCALC=2 default option, SURE automatically selects the most appropriate method.

3.3 *Tightness of the SURE Bounds*

This section informally shows why the SURE bounds are typically very close; it does that by presenting an intuitive formula for the relative difference between the bounds. SURE in no way depends on the arguments of this section, and the user need only look at the output of a run to see if the bounds are close for a particular problem. The relative difference between UB & LB is:

$$(\mathrm{UB}-\mathrm{LB})/(\mathrm{UB}) = [1 - Q(T-\Delta)/Q(T)]\cdot\left[\prod_{i=1}^{m} Z_i\right]\cdot\left[\prod_{j=1}^{n} Y_j/\mu(H_j)\right]$$

Y_j & Z_i are defined in section 3.2.

The fault arrival rates ϵ_i, α_j, β_j are on the order of $10-4$/hour and thus $\ll 1/T$; the $\mu(H_j)$ & $\sigma(H_j)$ are on the order of 10^{-4} hour and thus $\ll T$. Thus, $Y_j/\mu(H_j) \approx 1$ and $Z_i \approx 1$. Thus,

$$(\mathrm{UB}-\mathrm{LB})/\mathrm{UB} \approx 1 - Q(T-\Delta)/Q(T) \approx 1 - (1 - \Delta/T)^k$$

$$\approx k\cdot\Delta/T, \text{ for } \Delta \ll T.$$

For recovery times on the order of 10^{-4} hours, the r_i & s_j are on the order of 10^{-2} hours. Using fairly large values of Δ and k for $T = 10$ hours, viz, $\Delta = 0.1$ hours and $k = 5$:

$$(\mathrm{UB}-\mathrm{LB})/\mathrm{UB} \approx k\cdot\Delta/T = 5\cdot(0.1 \text{ hours})/(10 \text{ hours}) = 5\%.$$

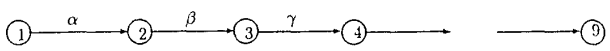
For smaller values of k and Δ, the relative error is obviously smaller, and as the failure rates or the recovery times decrease, the bounds tighten. From this we can see that the SURE bounds are tight for systems with slow fault arrivals and fast recoveries. These are precisely the characteristics of a well-designed fault-tolerant system.

4. PRUNING

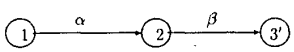
Although SURE can solve models with very many states, additional computational power is available by way of its pruning capability. The concept is simple.

Example 2

The following path is in a model:



Truncating the path at state 3 gives:



The following relationship exists between these models:

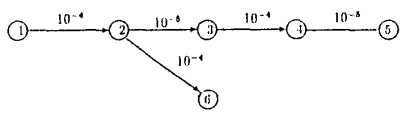
$$P_9(T) \leq P_{3'}(T).$$

Notation

$P_k(T)$ probability of reaching death-state k in time T □

Thus, the probability of reaching the new death-state 3′ is an upper bound on the probability of reaching state 9. SURE uses this strategy to prune long paths. However, SURE adds the *prune-state* probability to the upper bound so that a conservative answer is guaranteed. The prune-state probability is not added to the lower bound.

SURE uses a *prune-level* probability to determine which paths are pruned. As SURE traverses a path, it calculates the upper bound. If this upper bound falls below the prune-level probability, then the program prunes the rest of the path. There are two different ways to specify the prune-level probability – automatic and manual. The manual method requires the user to specify the prune-level probability with the PRUNE command, eg, PRUNE = 1E−12. This is illustrated in the session shown in figure 5.




```
% sure

  SURE V7.5   NASA Langley Research Center

  1? read prune.ex

  2: 1,2 = 1e-4;
  3: 2,3 = 1e-5;
  4: 2,6 = 1e-4;
  5: 3,4 = 1e-4;
  6: 4,5 = 1e-5;
  7:
  8: 11st=2;

  9? prune = 8e-11;
 10? run

DEATHSTATE    LOWERBOUND    UPPERBOUND    COMMENTS                    RUN #1
----------    ----------    ----------    ------------------------------------
        6     4.99650e-07   5.00000e-07
sure prune    0.00000e+00   1.66667e-11

TOTAL         4.99650e-07   5.00017e-07

 1 PATH(S) TO DEATH STATES, 1 PATH(S) PRUNED AT LEVEL  5.00000e-11
```

Figure 5. SURE Session Illustrating Pruning

The death-state 5 is never reached. SURE prunes this path and adds its *prune* probability to the state labeled sure *prune*. The system probability is obtained by adding all of the death-state contributions including the *sure prune* state. The automatic method leaves the setting of the prune level up to SURE. The program selects a prune level based on the probability of the first death-state it encounters. As more death-states are encountered, the program updates the value of PRUNE; it is updated after 10^0, 10^1, 10^2, ... death-states are reached.

5. SOLVING MODELS WITH LOOPS

Although the White bounding theorem deals only with paths that do not contain a loop, SURE uses a strategy based

upon pruning to solve models with loops. The difficulty with a model containing loops (its graph structure contains a circuit) is that there are an infinite number of paths. Consider the model in figure 6.

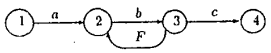


Figure 6. Model With A Loop

Unfolding the loop produces an infinite sequence of paths as shown in figure 7.

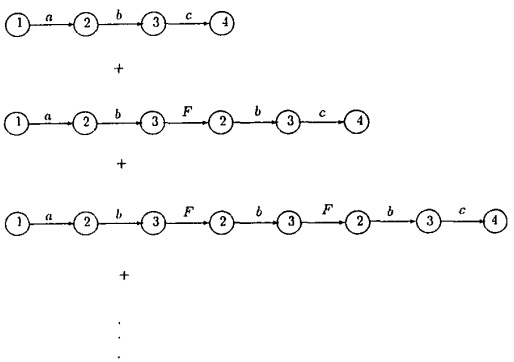


Figure 7. Infinite Sequence Of Paths

However, the situation is not intractable because the resulting sequence consists of increasingly longer paths. The death-state probabilities of the paths decrease rapidly like a Taylor's series. A typical sequence of probabilities is:

10^{-6}, 10^{-10}, 10^{-14}, 10^{-18}, 10^{-22}, ...

The series can be truncated at a point where the sum of all of the subsequent probabilities becomes negligible. SURE accomplishes this by a technique that enables a calculation of an upper and lower bound on the truncation error. The technique is based on the fact that a model with a loop is equivalent to a finite sequence of paths where only the last path in the sequence contains a loop. For example, the model of figure 6 can be reduced to the 3 paths shown in figure 8. The probability of entering the death-state of the original path is the sum of the probabilities of entering the death-state of the 3 new paths.

If the third path is *pruned* before the loop, then an upper bound can be obtained. Consequently, the sum of the 3 paths in figure 9 provides an upper bound on the original model of figure 6. Thus, a finite unfolding of a loop in conjunction with pruning can solve a model with loops. This is precisely the technique implemented in SURE.

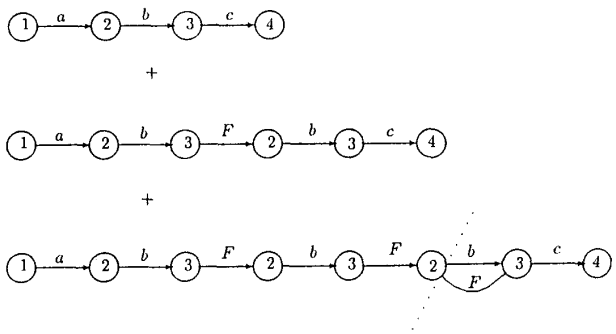


Figure 8. Reduction to a Finite Set of Paths

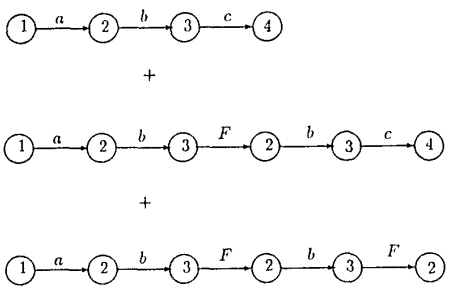


Figure 9. Model After Pruning and Reduction

6. OTHER SURE FEATURES

6.1 *Solving Models with Distributed Initial-State Probabilities*

Suppose that we need to solve a model whose initial state is not exactly determined. For example,

Pr{system is in state 1 at time 0} = 0.5,

Pr{system is in state 2 at time 0} = 0.3,

Pr{system is in state 1 at time 0} = 0.2.

SURE solves this model by the following strategy. The model is solved 3 times - first with state 1 as the start-state, then with state 2 as the start-state, and finally with the state 3 as the start-state. Each of these sub-results is multiplied by its start-state initial probability, then the resulting values are added together.

The SURE input language contains an INITIAL_P statement for initializing states. For example, INITIAL_P(1: 0.3, 2: 0.7); assigns an initial probability of 0.3 to state 1 and an initial probability of 0.7 to state 2. The user can specify upper and lower bounds on the initial state probabilities:

INITIAL_P(1: (0.27,0.31), 2: (0.69,0.71));.

6.2 *Operational State Probabilities*

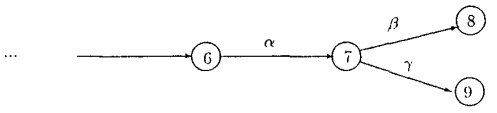


Figure 10. Submodel To Illustrate How SURE Calculates Operational State Probabilities

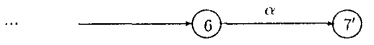


Figure 11. Reduced Submodel

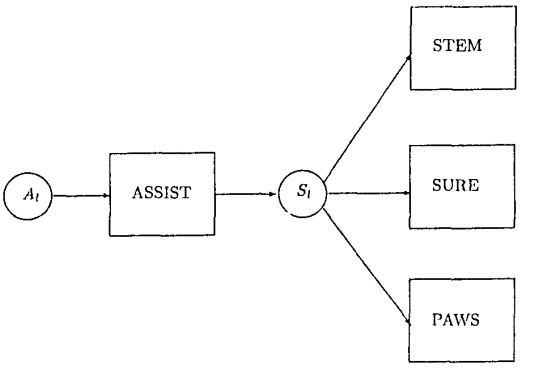


Figure 12. NASA Langley Reliability Analysis Workstation Concept

SURE can bound the operational state probabilities. The following method is used by the program. Given the submodel shown in figure 10, the program first calculates the death-state probabilities: $P_8(T)$ & $P_9(T)$. Next, the program solves the *reduced* model in figure 11, obtaining $P_{7'}(T)$. The probability of being in the operational state 7 in the original model is:

$$P_7(T) = P_{7'}(T) - [P_8(T) + P_9(T)].$$

Although this might seem laborious, it is efficiently implemented in SURE. All of the basic probabilities are calculated while SURE is traversing the graph of the model. Since the bounds are algebraic, the calculations are not costly. The subtractions are performed as SURE *backs out* of the recursions after reaching a death-state. The calculations are performed using bounds rather than exact probabilities. The bounds on the operational states are not as close as the bounds on the death-states, but are usually close enough to be useful (eg, for solving a sequence of semi-Markov models used for phased missions). The closer a state is to a death-state, the closer the bounds are.

6.3 *The ASSIST Front-End and Related Programs*

SURE is part of a reliability analysis workstation as illustrated in figure 12. This workstation concept is centered around the model-generation program, ASSIST (Abstract Semi-Markov Specification Interface to the SURE Tool). ASSIST processes a set of rules (A_l) that describe the construction of a model and then automatically generates the transitions of the model in the SURE input language (S_l). This language is overviewed in [17]; [18] describes this program in detail. Two additional programs, STEM & PAWS, have been developed as part of this workstation. These programs solve pure Markov models. They accept models in exactly the same input language as SURE, but solve the model using classical numerical techniques. PAWS uses *Páde* approximation with scaling, a technique developed by Ward [19]. STEM uses a Taylor's series technique. Both programs are described in detail in [20].

6.4 *Computation Time of SURE*

SURE has been distributed to over 60 industrial/academic sites and has been used on a wide range of problems. The computation time of the program varies appreciably with the structure and nature of the model. Complex systems requiring 10^5's of states have been solved. Nevertheless, some 10^4 state models require more execution time than some 10^5 state models. Table 1 provide an approximate view of the performance of SURE.

Table 1
SURE Execution Times for a Multiple-Triad System

Triads	States	Transitions	Execution Time
1	3	2	0.001
2	8	8	0.05
5	112	160	0.57
8	1280	2048	6.82
10	6144	10240	28.2
12	28672	49152	164
13	61440	106496	335

The first 3 columns give the number of items specified.
Execution times are in sec.

The models in table 1 describe systems of N-triads, varying N from 1 to 13. The models were generated using ASSIST and were solved using SURE V7.8 on a Sun Sparcstation 1+ with 24MB of RAM.

It is sometimes necessary to use a user-specified level of pruning to get good performance. Table 2 shows the potential benefit of such pruning. The execution times were obtained by solving a model of a triad with N warm spares, varying N from 1 to 200.

There are wide variations in SURE execution times depending upon the structure of the model and an effective choice of the prune level. In general, the performance of SURE is related more to the number of paths that must be traversed and their lengths rather than to the number of states.

Table 2
SURE Execution Times For Triad With Warm Spares

Spares	States	Transitions	Execution Time (auto-prune)	Execution Time (user-specified)
1	11	12	0.06	0.06
5	65	104	0.77	0.40
10	200	354	35.8	0.79
15	410	754	495	1.75
20	695	1304	3595	3.00
50	3980	7754	—	18.31
100	15455	30504	—	70.1
200	60905	121004	—	294.8

The first 3 columns give the number of items specified.
Execution times are in sec.

ACKNOWLEDGMENT

I am grateful to Dr. Alan White for the development of the mathematics upon which SURE is based. I greatly appreciate the many years of helpful advice and encouragement Dr. White has given to the development of SURE.

REFERENCES

[1] J. Stiffler, L. Bryant, L. Guccione, "CARE III final report phase I, volume I & II", NASA Contractor Report 159122 and 159123, 1979 Nov.
[2] K. S. Trivedi, J. B. Dugan, R. M. Geist, M. K. Smotherman, "Hybrid reliability modeling of fault-tolerant computer systems", *Int'l J. Computer & Electrical Engineering*, vol 2, 1985 Nov, pp 87-108.
[3] R. A. Sahner, K. S. Trivedi, "Reliability modeling using SHARPE", *IEEE Trans. Reliability*, vol R-36, 1987 Jun, pp 186-193.
[4] A. Costes, J. E. Doucet, C. Landrault, J. Laprie, "SURF: A program for dependability evaluation of complex fault-tolerant computing systems", in *Digest 11th Ann. Symp. Fault-Tolerant Computing*, 1981 Jun, pp 72-78.
[5] Y. Ng, A. Avizienis, "ARIES — An automated reliability estimation system", 1977 *Proc. Ann. Reliability and Maintainability Symp.*, pp 108-113.
[6] A. M. Johnson, M. Malek, "Survey of software for evaluating reliability, availability and serviceability", *ACM Computing Surveys*, vol 20, 1988 Dec, pp 227-270.
[7] A. L. White, "Reliability estimation for reconfigurable systems with fast recovery", *Microelectronics Reliability*, vol 26, no. 6, 1986, pp 1111-1120.
[8] A. L. White, "Synthetic bounds for semi-markov reliability models", NASA Contractor Report 178008, 1985.
[9] K. J. Dotson, "Examples of non-conservatism in the CARE III program", NASA Technical Memorandum 100524, 1988 Jan.
[10] D. M. Rose, R. E. Altschul, J. W. Manke, D. L. Nelson, "Review and verification of CARE III mathematical model and code: Interim report", NASA Contractor Report 166096, 1983 Apr.
[11] US Department of Defense, *Reliability Prediction of Electronic Equipment*, 1982 Jan. Mil-Hdbk-217D.
[12] D. P. Siewiorek, R. S. Swarz, *The Theory and Practice of Reliable System Design*, 1982; Digital Press.
[13] J. H. Lala, T. B. Smith III, "Development and evaluation of a fault-tolerant multiprocessor (FTMP) computer, vol III — FTMP test and evaluation", NASA Contractor Report 166073, 1983.
[14] R. M. Geist, K. S. Trivedi, "Ultrahigh reliability prediction in fault-tolerant computer systems", *IEEE Trans. Computers*, vol C-32, 1983, pp 1118-1127.
[15] R. W. Butler, A. L. White, "SURE reliability analysis: Program and mathematics", NASA Technical Paper 2764, 1988 Mar.
[16] A. L. White, "Upper and lower bounds for semi-Markov reliability models of reconfigurable systems", NASA Contractor Report 172340, 1984.
[17] R. W. Butler, "An abstract language for specifying markov reliability models", *IEEE Trans. Reliability*, vol R-35, 1986 Dec, pp 595-601.
[18] S. C. Johnson, "ASSIST user's manual", NASA Technical Memorandum 87735, 1986 Aug.
[19] R. C. Ward, "Numerical accuracy of the matrix exponential with accuracy estimate", *SIAM J. Numerical Analysis*, vol 14, 1977, pp 600-610.
[20] R. W. Butler, P. H. Stevenson, "The PAWS and STEM reliability analysis programs", NASA Technical Memorandum 100572, 1988 Mar.
[21] R. W. Butler, S. C. Johnson, "The art of fault-tolerant system reliability modeling", NASA Technical Memorandum 102623, 1990 Mar.

AUTHOR

Ricky W. Butler; MS 130; NASA Langley Research Center; Hampton, Virginia 23665 USA.

Ricky W. Butler is a senior research engineer at the NASA Langley Research Center. He received his BA from the University of Virginia in Mathematics in 1976 and his MS in Computer Science from the University of Virginia in 1978. His research interests are in design and validation of fault-tolerant computer systems used for flight-critical applications.

Manuscript TR90-193 received 1990 October 17; revised 1991 July 29.

IEEE Log Number 03301 ◄TR►

FROM THE EDITORS FROM THE EDITORS FROM THE EDITORS FROM THE EDITORS FROM THE EDITORS FROM THE EDITORS

Change: To a Quarterly

Effective with the 1992 March issue, this *Transactions* has returned to a quarterly publication. In 1973, this *Transactions* changed from a quarterly to 5 issues per year for two reasons: 1) it improved the net income for the *Transactions*, and 2) it provided a space (February) for another publication, *Proceedings of the Annual Reliability & Maintainability Symposium*, to be sent to our full members. Neither of those situations is now the same; thus the publication schedule is returning to quarterly. The budgeted, annual number of pages is staying the same. ◄TR►